11 Other Applications

...

The complete program in K is (the order of rules matters):

import $INT,\ K\text{-}BASIC$
$k : Comp \to ConfigItem\ [struct]$
$i,\ c,\ d : Int \to ConfigItem\ [struct]$
$Int < CompItem$
$q : Int \to Int$
$r : Config \to Int$

$$q(N) = r(k(0)\ i(N)\ c(0)\ d(1))$$
$$r\langle k(\cdot)\ c(I)\rangle = I$$

$\{\_,\_,\_\} : Int \times Comp \times Int \to CompItem\}$ ...

$$k(\underset{\cdot}{\underline{N}}\rangle\ i(N)\ d(\underset{D-1}{\underline{D}})$$
$$k(\underset{\{I,K,1\}}{\underline{I}} \curvearrowright K)$$
$$k(\underset{\cdot}{\underline{\{\_,\cdot,\_\}}}\rangle\ i(N)\ c(\underset{J+1}{\underline{J}})\ d(\underset{N-1}{\underline{N}})$$
$$k(\underset{0 \curvearrowright I}{\underline{\{I,\cdot,\_\}}}\rangle\ d(\underset{D+1}{\underline{D}})$$
$$\{I, J \curvearrowright K, M\} = \text{if } |I-J| \in M\ 0 \text{ then } I \text{ else } \{I, K, M+1\}$$

The Maude-ified version of the K program above is:

```
fmod M is
  protecting INT .

  sort Configuration .
  op empty : -> Configuration .
  op __ : Configuration Configuration -> Configuration [assoc comm id: empty] .

  sorts Computation ComputationItem .
  subsorts Int ComputationItem < Computation .
  op nil : -> Computation .
  op _->_ : Computation Computation -> Computation [assoc id: nil] .

  op q : Nat -> Nat .
  op r : Configuration -> Nat .
  vars I J N M D : Int .  var K : Computation .

  op k : Computation -> Configuration .
  ops i c d : Int -> Configuration .

  eq q(N) = r(k(0) i(N) c(0) d(1)) .
  eq r(k(nil) i(N) c(I) d(D)) = I .

  eq k(N -> K) i(N) d(D) = k(K) i(N) d(D - 1) .
  eq k(I -> K) i(N) d(M) = k([I + 1, K, 1] -> K) i(N) d(M) [owise] .
  eq k([I,nil,M] -> K) i(N) c(J) d(N) = k(K) i(N) c(J + 1) d(N - 1) .
  eq k([I,nil,M] -> K) i(N) c(J) d(D) = k(0 -> I -> K) i(N) c(J) d(D + 1) [owise] .

  op [_,_,_] : Int Computation Int -> ComputationItem .
  eq [I, J -> K, M] = if I == J or I - J == M or J - I == M then I else [I, K, M + 1] fi .
endfm
```